AUDITING DATA USING OBSERVABLE
AND OBSERVER OBJECTS

APPLICATION FOR

UNITED STATES LETTERS PATENT

"Express Mail" Mailing Label No.:   ET251817955US

Date of Deposit: May 3, 2001

I hereby certify that this paper is being
deposited with the United States Postal Service as
"Express Mail Post Office to Addressee" service
under 37 CRF 1.10 on the date indicated above and is
addressed to Box Patent Application, Assistant
Commissioner for Patents, Washington, D.C. 20231.

Name: Susan L. Phelps

Signature: _____

INTERNATIONAL BUSINESS MACHINES CORPORATION

AUDITING DATA USING OBSERVABLE
AND OBSERVER OBJECTS

Technical Field

[0001]   This invention relates, in general, to data audit
and verification, and in particular, to using decentralized
objects to audit the data and to handle changes to the data.

Background Art

[0002]   Data verification can be a complex and tedious
task requiring the use of complicated conditional logic.
For example, in order to verify the fields of a data entry
form, each field has to be verified. Since each field has
its own rules for verifying or auditing its entered data,
conditional logic (e.g., if-then-else statements) is needed
for each and every field to be audited. As the number of
fields to be audited increases, so does the complexity of
the conditional logic. Similarly, as the number of
conditions to be tested for each field increases, so does
the complexity of the conditional logic.

[0003]   Previously, the conditional logic used for the
various fields of a form has been combined into a central
location (e.g., an application) that runs through the logic
serially. This suffers from complexity, as well as
performance degradation.

[0004]   Based on the foregoing, a need exists for a
capability that facilitates the auditing and verifying of
fields of a data entry form. Further, a need exists for a
capability that facilitates the auditing and verifying of
other features. In particular, a need exists for a
capability that replaces the complex and confusing
conditional logic typically used to verify or audit data. A
need exists for a capability that decentralizes and
automates the verification process.

Summary of the Invention

[0005]   The shortcomings of the prior art are overcome and
additional advantages are provided through the provision of
a method of auditing data of data entry forms. The method
includes, for instance, providing an observable object for a
field of a data entry form to be audited, the observable
object including logic to be used for auditing data of the
field; and auditing data of the field using the observable
object.

[0006]   In a further aspect of the invention, a method of
auditing data of a data entry form is provided. The method
includes, for instance, providing an observable object for a
field of the data entry form to be audited, the observable
object including logic to be used for auditing data of the
field; auditing data of the field using the observable
object; building a modifier object for the field, when the
auditing indicates that a change has occurred in the data;
forwarding the modifier object to a pool of one or more

modifier objects; informing an observer that the modifier
object has been added to the pool of one or more modifier
objects; retrieving, by the observer, the modifier object
from the pool; and running, by the observer, a modify method
of the modifier object to accommodate the change in the
data.

[0007]   In yet a further aspect of the present invention,
a method of auditing data of components of a self-monitoring
framework is provided. The method includes, for instance,
providing an observable object for each component of a
plurality of components of multiple components of the self-
monitoring framework to be audited, each observable object
including logic to be used for auditing data of its
associated component; and auditing data of the each
component using the observable object corresponding to that
component.

[0008]   System and computer program products corresponding
to the above-summarized methods are also described and
claimed herein.

[0009]   Advantageously, a capability is provided that
facilitates the auditing of data. In one example, the
capability provides a decentralized and automated technique
for providing data verification. The capability
advantageously combines the Model-View-Control object-
oriented methodology with observable objects and modifier
object pooling for observer auditing. This provides a

simplified and straight forward capability for auditing data that is flexible and extendable.

[0010]   Additional features and advantages are realized through the techniques of the present invention. Other embodiments and aspects of the invention are described in detail herein and are considered a part of the claimed invention.

Brief Description of the Drawings

[0011]   The subject matter which is regarded as the invention is particularly pointed out and distinctly claimed in the claims at the conclusion of the specification. The foregoing and other objects, features, and advantages of the invention are apparent from the following detailed description taken in conjunction with the accompanying drawings in which:

[0012]   FIG. 1 depicts one example of a computing environment incorporating and using aspects of the present invention;

[0013]   FIG. 2 depicts one embodiment of a data entry form including one or more fields to be audited, in accordance with an aspect of the present invention;

[0014]   FIGs. 3a-3b depict examples of conventional conditional logic used to audit the fields of FIG. 2;

[0015]   FIG. 4 depicts one embodiment of hardware units to be audited, in accordance with an aspect of the present invention;

[0016]   FIG. 5 depicts one example of an observable object used in auditing the Customer_Id field of FIG. 2, in accordance with an aspect of the present invention;

[0017]   FIG. 6a depicts one embodiment of the logic associated with using an observable object to audit a feature, in accordance with an aspect of the present invention;

[0018]   FIG. 6b depicts one embodiment of the logic used to accommodate an event associated with an observable object, in accordance with an aspect of the present invention;

[0019]   FIG. 7 depicts one example of a modifier object created for the Customer_Id field of FIG. 2, in accordance with an aspect of the present invention;

[0020]   FIG. 8 depicts an overview class diagram of a model used to audit a feature and to effect

necessary changes, in accordance with an aspect of the present invention; and

[0021]   FIG. 9 depicts an overview diagram of the objects associated with auditing the hardware units of FIG. 4, in accordance with an aspect of the present invention.

Best Mode for Carrying Out the Invention

[0022]   In accordance with an aspect of the present invention, data is audited using observable objects, which include auditing logic. When the auditing logic of a particular observable object detects that a specified event, such as a change in the data, has occurred, then a modifier object is automatically created, which includes the logic for accommodating the specified event. The modifier object is added to a pool of objects, and it is the responsibility of an observer to retrieve the modifier object and execute its logic. The data to be audited can be associated with a variety of features, including but not limited to, fields within data entry forms or components of a self-monitoring framework.

[0023]   One example of a computing environment incorporating and using aspects of the present invention is depicted in FIG. 1. In one example, a computing environment 100 includes, for instance, at least one central processing unit (CPU) 102, a main storage 104, and one or more input/output devices 106, each of which is described below.

[0024]   As is known, central processing unit 102 is the controlling center of the computing environment and provides the sequencing and processing facilities for instruction execution, interruption action, timing functions, initial program loading and other machine related functions. The central processing unit executes at least one operating system used to control the operation of the computing environment by controlling the execution of other programs, controlling communication with peripheral devices and controlling use of the computer resources.

[0025]   Central processing unit 102 is coupled to main storage 104, which is directly addressable and provides for high-speed processing of data by the central processing unit. Main storage 104 may be either physically integrated with the CPU or constructed in stand-alone units.

[0026]   Main storage 104 is also coupled to one or more input/output devices 106. These devices include, for instance, keyboards, communications controllers, teleprocessing devices, printers, magnetic storage media (e.g., tape, disks), direct access storage devices and sensor-based equipment. Data is transferred from main storage 104 to input/output devices 106, and from the input/output devices back to main storage.

[0027]   In one example, computing environment 100 is a single system environment, which includes an RS/6000 computer system running an AIX operating system. (RS/6000 and AIX are offered by International Business Machines

Corporation.) The invention is not limited to such an environment, however. The capabilities of the present invention can be incorporated and used within many types of computing environments and many types of computer systems. For instance, computing environment 100 can include a UNIX workstation running a UNIX-based operating system. Other variations are also possible and are considered a part of the claimed invention.

[0028]   In yet another embodiment, computing environment 100 is a multisystem environment in which various computing units are coupled together via a connection, such as a wire connection, a token ring or network connection, to name just a few examples. Further, the computing environment may include a large parallel system with a plurality of units coupled to one another via a network connection, such as a switch. Again, the capabilities of the present invention are usable with many types of computing environments, as well as other types of communications systems.

[0029]   Various features, including features of a computing environment, can be audited or verified, in accordance with an aspect of the present invention. For example, a data entry form may include one or more fields to be audited. That is, data associated with a field (e.g., input data or provided data) is verified. If the data has changed, then one or more specified actions are taken.

[0030]   As a further example, the features may include a plurality of components of a self-monitoring framework.

For example, one or more operating conditions of each component of the framework to be monitored are audited, and one or more specified actions are taken, if a specified event occurs.

[0031]   Although data entry fields and components are provided herein as examples of features to be audited, the invention is not limited to those features. Aspects of the present invention are equally applicable to other features (which may or may not be associated with a computing environment), and those are therefore, considered within the scope of the present invention. Further details of the invention are described with reference to the remaining figures. Again, the examples focus on fields of a data entry form and components of a self-monitoring framework, but aspects of the invention are not limited to such examples.

[0032]   An example of a data entry form is depicted in FIG. 2. A data entry form 200 includes a plurality of features 202. In this example, the features are fields within the data entry form, and each field includes data that is to be audited. For example, a Customer_Id field includes data 204 that is to be audited. Likewise, a Name field includes data 206 that is to be audited, and so forth. Each of these fields typically has its own set of rules to follow. Thus, traditionally, in order to audit the data of each field, conditional logic (such as, if-then-else statements) specific to the field has been used to perform the audit (see, e.g., FIG. 3a).

[0033]   As shown in FIG. 3a, the logic for each field is centrally located as part of an application that is responsible for performing the auditing. Thus, if there are ten fields, then there are ten conditional statements within the application for checking the fields of the data form. Each conditional statement has one or more conditions to test.

[0034]   As the number of fields and/or the number of conditions to be tested increases, the more complex the conditional logic, as shown in FIG. 3b.

[0035]   In another embodiment, the features to be audited include components of a self-monitoring framework. In one example, each component is a hardware unit (e.g., a machine, a node of a computing environment, and/or other types of units) 400 (FIG. 4). The hardware units are coupled to one another, and one or more of those units are to be audited. The auditing includes checking one or more operating conditions of each unit to be audited. The operating condition(s) for each unit may be the same or different.

[0036]   Previously, a central monitor 402 has been used to monitor the operating conditions of each unit. That is, the monitor is part of a monitoring loop that starts at a specified location and goes around the loop testing the operating or specified conditions. Thus, the monitor includes the auditing logic for each of the units to be monitored, which leads to complexity, reliance on a single monitoring component, and increased communication overhead.

[0037]   In order to overcome deficiencies of previous
techniques associated with auditing various features,
including reducing the complexity of the auditing logic and
decentralizing the auditing responsibility, each feature to
be audited is represented by an observable object, in
accordance with an aspect of the present invention. An
example of an observable object is depicted in FIG. 5. In
FIG. 5, an observable object 500 corresponds to the
Customer_Id field of FIG. 2. There would be one such
observable object for each field or feature to be audited.
Each observable object has its own audit method 502, which
reads the data associated with its corresponding feature and
performs the auditing, checking and verifying, as
appropriate. No other object knows about the data of this
object's feature. The observable object provides a
decentralized location for the auditing logic, which is
associated with its particular feature to be audited.

[0038]   One embodiment of the logic associated with using
observable objects to audit a feature is described with
reference to FIGs. 6a and 6b. In one example, this logic is
implemented using object-oriented programming concepts, such
as available with JAVA. However, other object-oriented
concepts, as well as other programming concepts, may be used
to implement aspects of the present invention.

[0039]   Referring to FIG. 6a, in order for a feature to be
audited, initially, an observable object is created for the
feature, STEP 600. The manner in which an observable object
is created is known in the art. However, information

regarding observable objects can be found, for instance, at the URL of http://java.sun.com/j2se/1.3/docs/api/index.html, the contents of which are hereby incorporated herein by reference in their entirety.

[0040]   Subsequent to creating the observable object for the particular feature, the observable object is used to perform an audit, STEP 601. The audit includes checking or verifying aspects of the data, as desired. If the audit should fail, an error message is placed in the status area. Otherwise, processing continues.

[0041]   In particular, a determination is made as to whether a specified event has occurred, INQUIRY 602. For example, if the feature is a field, a determination is made as to whether the data of the particular field associated with this observable object has changed, as indicated by the audit. If the specified event has occurred, then a modifier object is automatically generated, STEP 604.

[0042]   One example of a modifier object 700 is depicted in FIG. 7. The modifier object is instantiated from a class extending a modifier interface. Hence, each modifier object has a modify method 702. The modify method includes the logic to accommodate the event that has occurred. For example, if a change has been made to the data of a field, then that change is propagated to a storage medium, such as a database. Other or different actions can also be taken.

[0043]   Referring back to FIG. 6a, subsequent to generating the modifier object, the modifier object is added to a pool of one or more modifier objects, STEP 606. As examples, this may include adding the modifier object to a queue, a stack, a linked list or a hash table. This allows one or more observers of the observable object to be notified that the object has been added to the pool, STEP 608. An observer is, for instance, an object that implements an interface observer. An observer is notified, for example, by the observable's notifyObservers method.

[0044]   Returning to INQUIRY 602, if a specified event has not occurred, then no modifier object is created and the process is complete.

[0045]   The logic of FIG. 6a is performed for each feature to be audited. That is, each feature has its own observable object with its own auditing logic.

[0046]   Referring now to FIG. 6b, when an observer receives notification that the object has been added to the pool, STEP 612, the observer has the ability to retrieve the modifier object from the pool, STEP 614. When the observer retrieves the modifier object from the pool, the observer runs the modify method to accommodate the event that has taken place, STEP 616. For example, changed data is written to a database.

[0047]   Subsequently, in one example, the observer determines whether there are more modifier objects in the

pool, INQUIRY 618. If so, then another modifier object is retrieved, STEP 614. Otherwise, the processing ends. (In another embodiment, the observer retrieves only one object at each notification.)

[0048]   In one embodiment, a plurality of observers are notified. Thus, the observers collaborate with one another, in order to avoid executing the modify method multiple times. However, in another embodiment, no such collaboration may be desired.

[0049]   Further, in one example, if there is a certain order to be followed in the processing of the observer objects, then some scheme of number association can be tagged to each modifier object. This enables the observer to retrieve the objects from the pool in the specified order.

[0050]   The observer can retrieve the modifier object and run its modify method anytime after the retrieval. Further, in one embodiment, the observer can wait for a plurality of modifier objects to be placed in the pool before it starts to retrieve the objects and run the modify methods.

[0051]   An Overview class diagram of the observable/ observer model used for aspects of the present invention is depicted in FIG. 8. As shown in FIG. 8, a user interface panel 800 includes one or more fields 802. For each field, there is an observable object 804. The observable object includes a notify method 806 to notify any registered

observers of a specified event, and an audit method 808 to
check the data of the field. Should a specified event be
detected by audit method 808, then a modifier object 810 is
generated. The modifier object includes a modify method
812. This method may include a store and/or retrieval 814
of the data to effect the change.

[0052]   Further, an observer 816 is registered to the
observable object. When the modifier object is added to the
pool of objects, the observable object notifies the
observer. As a further example, a submit button 818 may be
used by a user to perform the notification.

[0053]   Another overview diagram is depicted in FIG. 9,
which corresponds to the example in which the feature is a
component. As depicted in FIG. 9, each component (e.g., a
unit or another component, in a further example) to be
audited has an observable object 900. Each observable
object is used to audit its corresponding component. As an
example, each observable object tests one or more operating
conditions that are appropriate for its component. If the
audit determines that a specified event for a particular
unit has occurred, then a modifier object 902 is generated
for its corresponding observable object. Further, each
modifier object is added to a pool 904.

[0054]   Moreover, an observer 906 (or a plurality of
observers) is notified that modifier objects have been added
to the pool. This allows the observer to retrieve objects
from the pool.

[0055]   As described above, in accordance with an aspect of the present invention, an observable object is created for each feature to be audited. In a further embodiment, since each feature is represented by an observable object, a separate observer can be written for testing purposes. This test observer is notified when testing is to be performed. The test observer can perform actions, such as, for instance, displaying the audited data; displaying more detailed information about data entered and what had been changed, for debugging purposes; as well as other actions.

[0056]   The test observer runs a test method, which is included in a modifier object. That is, in one example, each modifier object has a test method, which provides a mechanism for other types of testing of the data, such as, for instance, value ranges; invalid values; and boundary values. The test observer runs the test method, instead of the modify method, to perform the testing. This allows the capabilities of the present invention to be extendable for testing and debugging.

[0057]   By having each feature build its own modifier object, the audit requirements are encapsulated in the object (i.e., in the modified method). The observer, in turn, only activates the changes (e.g., changing of data in storage), according to the results of individual audits. In other words, the central point of auditing logic is now distributed into the individual objects.

[0058]   Thus, one observer can handle a large range of features with one general process. Further, multiple observers can observe the same features. Yet further, many features can be divided and observed by different observers. This provides flexibility and scalability.

[0059]   Described in detail above is a model that uses observable/observer objects and object pooling techniques to audit features. The model provides a decentralized technique in which the auditing logic is distributed. This eliminates the complex and confusing ad-hoc if-then-else logic currently used. Further, it enables the auditing and observing of many features concurrently and automatically.

[0060]   The present invention can be included in an article of manufacture (e.g., one or more computer program products) having, for instance, computer usable media. The media has embodied therein, for instance, computer readable program code means for providing and facilitating the capabilities of the present invention. The article of manufacture can be included as a part of a computer system or sold separately.

[0061]   Additionally, at least one program storage device readable by a machine, tangibly embodying at least one program of instructions executable by the machine to perform the capabilities of the present invention can be provided.

[0062]   The flow diagrams depicted herein are just examples. There may be many variations to these diagrams or

the steps (or operations) described therein without departing from the spirit of the invention. For instance, the steps may be performed in a differing order, or steps may be added, deleted or modified. All of these variations are considered a part of the claimed invention.

[0063]   Although preferred embodiments have been depicted and described in detail herein, it will be apparent to those skilled in the relevant art that various modifications, additions, substitutions and the like can be made without departing from the spirit of the invention and these are therefore considered to be within the scope of the invention as defined in the following claims.